Abnormal data flow inspection apparatus for use during program debugging - has controller which regulates execution of source program based on output from inspection unit

Patent Assignee: NIPPON STEEL CORP (YAWA)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Main IPC	Week
JP 10289124	A	19981027	JP 9832613	A	19980216	G06F-011/28	199902 B

Priority Applications (No Type Date): JP 9732415 A 19970217
Patent Details:

Patent	Kind	Lan	Pg	Filing Notes	Application	Patent
JP 10289124	A		9			

Abstract (Basic): JP 10289124 A

The apparatus has an inspection unit (24) which performs search of path in which abnormalities arises in a data flow in a source program (20). The number of paths searched is estimated by an estimation unit (22). A controller (30a) regulates execution of source program based on output from inspection unit.

ADVANTAGE - Increases work efficiency by detecting abnormal data flow quickly. Improves operativity by eliminating need for operator to indicate each execution.

Dwg.6/7

Title Terms: ABNORMAL; DATA; FLOW; INSPECT; APPARATUS; PROGRAM; DEBUG; CONTROL; REGULATE; EXECUTE; SOURCE; PROGRAM; BASED; OUTPUT; INSPECT; UNIT
Derwent Class: T01
International Patent Class (Main): G06F-011/28
International Patent Class (Additional): G06F-009/06
File Segment: EPI
Manual Codes (EPI/S-X): T01-F06; T01-G05A

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－289124

(43)公開日　平成10年(1998)10月27日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　11/28		Ｇ０６Ｆ　11/28	Ａ
9/06	５３０	9/06	５３０Ｈ
	５４０		５４０Ｕ

審査請求　未請求　請求項の数５　ＯＬ　（全９頁）

(21)出願番号　特願平10－32613

(22)出願日　平成10年(1998)２月16日

(31)優先権主張番号　特願平9－32415

(32)優先日　平９(1997)２月17日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000006655
新日本製鐵株式会社
東京都千代田区大手町２丁目６番３号

(72)発明者　木村　博昭
東京都千代田区大手町２丁目６番３号　新日本製鐵株式会社内

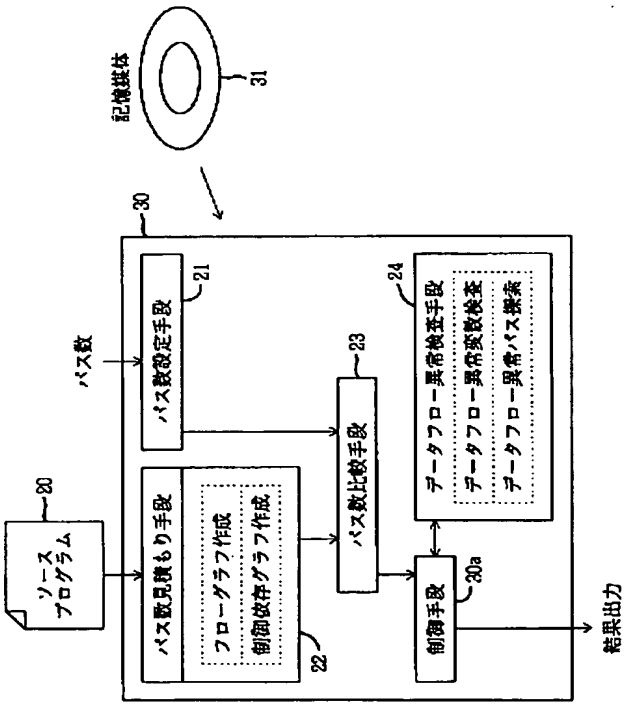
(74)代理人　弁理士　半田　昌男

(54)【発明の名称】　データフロー異常検査装置

(57)【要約】

【課題】　データフロー異常の検査を実行する際に自動的にそのソースプログラムのパス数を見積もって、データフロー異常の検査においてパスの探索も行うかどうかの判断を行いやすくするものである。

【解決手段】　検査対象であるソースプログラムを読み込ませたら、まず、そのパス数を計算で求め、この求めたパス数に基づいて、これ以降の動作を制御する。パス数が分かれば、そのソースプログラムについて行うデータフロー異常の検査に要する時間を予測できる。パス数を計算するめに、まず、読み込んだソースプログラムに基づいてフローグラフを作成する。フローグラフは、ソースプログラムに含まれるif文のところで、パスが二つに分かれるように描かれる。フローグラフが描かれたら、制御依存グラフを作成し、この制御依存グラフを用いて所定の計算を行うことによって、パス数を見積もる。



1

【特許請求の範囲】

【請求項１】　ソースプログラムに対してデータフロー異常の検出と、そのようなデータフロー異常が起こるパスの探索とを行うデータフロー異常検査手段と、

前記データフロー異常の探索を行うためのパス数を見積もるパス数見積もり手段と、

前記パス数見積もり手段によって得られたパス数に基づいて、前記検査手段による前記ソースプログラムに対するデータフロー異常の検査の実行を制御する制御手段と、

を具備することを特徴とするデータフロー異常検査装置。

【請求項２】　前記制御手段は、前記パス数が所定の値以下の場合には前記データフロー異常の検出及びデータフロー異常が起こるパスの探索を行うように、また、前記パス数が前記所定の値を超える場合には前記データフロー異常の検出は行うがデータフロー異常が起こるパスの探索は行わないように、前記データフロー異常検査手段を制御することを特徴とする請求項１記載のデータフロー異常検査装置。

【請求項３】　前記制御手段は、前記パス数見積もり手段によって前記パス数が得られたら、動作を一時停止し、そのパス数をユーザーインターフェースでオペレーターに知らせ、その後の動作の続行をオペレータの判断に委ねることを特徴とする請求項１記載のデータフロー異常検査装置。

【請求項４】　前記パス数見積もり手段は、前記ソースプログラムに対応したフローグラフを作成し、前記フローグラフに基づいて制御依存グラフを作成し、前記制御依存グラフに基づいて所定の計算を行って前記パス数を見積もることを特徴とする請求項１，２又は３記載のデータフロー異常検査装置。

【請求項５】　コンピュータを、

ソースプログラムに対してデータフロー異常の検出と、そのようなデータフロー異常が起こるパスの探索とを行うデータフロー異常検査手段と、

前記データフロー異常の探索を行うパスのパス数を見積もるパス数見積もり手段と、

前記パス数見積もり手段によって得られたパス数に基づいて、前記検査手段による前記ソースプログラムに対するデータフロー異常の検査の実行を制御するための手段として、

機能させるためのプログラムを記録した機械読み取り可能な記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、作成したコンピュータプログラムのデバッグ作業を行う場合などに使用されるデータフロー異常検査装置に関する。

【０００２】

2

【従来の技術】コンピュータプログラムの作成は、人間の手によって行われる以上、種々の欠陥が生じうる。その一つに、データフロー異常がある。コンピュータプログラムのソースコードの中に異常なデータフローがあっても、これをコンパイルする際には検出されず、そのまま機械語のプログラムに翻訳される。しかし、このようなプログラムを実行すると、特定のパスを通る動作をするときに、実行が停止し、動作しなくなる場合がある。

【０００３】かかるデータフローの異常を予め検出して、プログラムから排除するための装置として、データフロー異常検査装置が知られている。このデータフロー異常検査装置は、一般にコンピュータに搭載した専用のソフトウェアによって実現される。作成したソースプログラムを予めデータフロー異常検査装置にかけて、データフローの異常を検出し、排除するという作業を行うことによって、作成されたプログラムの信頼性及び品質を向上させることができ、また、そのプログラムの稼働を開始した後に、あるいはそのプログラムを含んだ製品を出荷した後に、データフロー異常に基づく重大な欠陥が発生するという可能性が軽減される。

【０００４】

【発明が解決しようとする課題】従来のデータフロー異常検査装置は、検査対象となるソースプログラムが入力されると、「データフロー異常の検出」と「データフロー異常が起こるパスの探索」という二つの作業を行う。「データフロー異常の検出」とは、データフロー異常がプログラム上のどこに発生しているかを探し出す作業である。一方、「データフロー異常が起こるパスの探索」とは、データフロー異常がどのようなパスを通ったときにそれが発生するのかを知るために、ソースプログラムの中のパスを調べる作業である。

【０００５】ところで、データフロー異常検査装置によってソースプログラムのデータフロー異常を検査する場合、その検査対象とされるプログラムの規模が大きくなるに従ってデータフロー異常の検査に要する時間が急激に増大する。これは、データフロー異常検査のうち、特にデータフロー異常が起こるパスの探索を行う際にプログラム中にあるすべてのパスを辿るため、通常プログラムの規模が大きくなるとパスの数も急激に多くなるからである。一般的なワークステーション上で行うこのようなパスの探索は、プログラムの規模によっては数日から１週間程度かかる場合がある。

【０００６】このようにパスの探索に長い時間がかかると、その間プログラマーはいつになったらデータフロー異常の検査が終了するのかを知ることができず、ワークステーションがそのデータフロー異常の検査のためだけに専有された状態で、作業が終了するまで待つしかなかった。また、このようにデータフロー異常の検査に要する時間が長くかかると、途中でそのワークステーションを他の用途に使用する必要が生じた場合には、途中でデ

ータフロー異常の検査を中止しなければならず、それまでに費やした時間が無駄になることがあった。

【０００７】本発明は、上記事情に基づいてなされたものであり、データフロー異常が起こるパスの探索を実行する際に自動的にそのソースプログラムのパス数を見積もって、データフロー異常の検査においてパスの探索も行うかどうかの判断を行いやすくすることができるようにしたデータフロー異常検査装置を提供することを目的とするものである。

【０００８】

【課題を解決するための手段】上記の課題を解決するために、本発明は、ソースプログラムに対してデータフロー異常の検出と、そのようなデータフロー異常が起こるパスの探索とを行うデータフロー異常検査手段と、前記データフロー異常の探索を行うためのパス数を見積もるパス数見積もり手段と、前記パス数見積もり手段によって得られたパス数に基づいて、前記検査手段による前記ソースプログラムに対するデータフロー異常の検査の実行を制御する制御手段とを具備することを特徴とする。

【０００９】本発明は、上記より、データフロー異常の検査を行う際に、検査対象であるソースプログラムのパス数を予め見積もり、このパス数によって、データフロー異常が起こるパスの探索を実行するかどうかを制御するので、例えば、データフロー異常の検査を実行する前に、その検査にかかる大体の時間が分かり、したがって、探索に非常に長い時間がかかることが分かった場合には探索を中止するなど、それぞれのソースプログラムについて柔軟な対応が可能となる。

【００１０】

【発明の実施の形態】以下に図面を参照して、本発明の一実施形態について説明する。まず、データフロー異常について簡単に説明する。尚、ここでは、Ｃ言語で記述されたソースプログラムに基づいて説明するが、他の言語の場合も同様に考えることができる。一般に、ソースプログラムの中では、データに対しては、「定義（ｄ）」、「未定義（ｕ）」、「参照（ｒ）」という三つのイベントが発生しうる。ここで、「定義（ｄ）」には、データ宣言による明示的な定義あるいは“ａ＝ｂ”の左辺“ａ”に相当する代入式の左辺値として現れる定義、ファイルのオープンなどがある。「未定義（ｕ）」には、参照不可能になった場合あるいは内容が不定になった場合がある。そして、「参照（ｒ）」は、主に、計算の中で参照される場合と述語の中で参照される場合がある。

【００１１】すべてのデータは、「定義した上で参照し、最後に未定義にする」という一連の順序に従って状態が遷移するように使用しなければならない。あるデータのフローがこの規則に従っているかどうかを調べるために、そのデータのある時点での状態と、これに対するイベントを組み合わせたもの（これを「基本フロー」と称する。）に分解して考えることができる。図１は、データフローを状態遷移図として示した図である。この図で、各ノードは、プログラムによって指定されるデータの状態を示し、Ｕは未定義、Ｄは定義、Ｒは参照、Ａはデータフロー異常である。また、各矢印は、ある状態が上記のイベントによって、別の状態（又は同じ状態）へ遷移することを示している。

【００１２】図１に示すように、Ｕの状態にあるデータに対してｄというイベントが発生すると（これを基本フロー「Ｕｄ」と書く。以下同様。）、そのデータの状態はＤとなる。Ｕの状態にあるデータに対して、ｒ又はｕというイベントが発生すると、そのデータはデータフロー異常Ａとなる。また、Ｒという状態にあるデータに対して、ｕというイベントが発生すると、そのデータの状態はＵとなる。Ｒという状態にあるデータに対して、ｄというイベントが発生すると、そのデータの状態はＤとなる。Ｒという状態にあるデータに対して、ｒというイベントが発生すると、そのデータの状態はＲのままである。Ｄという状態にあるデータに対して、ｒというイベントが発生すると、そのデータの状態はＲとなる。Ｄという状態にあるデータに対して、ｄ又はｕというイベントが発生すると、そのデータの状態はデータフロー異常Ａとなる。このように、基本フローは、全部で九種類ある。尚、データフロー異常の状態にあるデータに対しては、どのようなイベントが発生してもデータフロー異常のままであるが、かかる状態は基本フローには含めない。

【００１３】この九種類の基本フローの中で、Ｄｒ，Ｕｄ，Ｒｕ，Ｒｒ，Ｒｄの五種類は正しい基本フローと考えられる。一方、Ｕｒ，Ｕｕ，Ｄｕ，Ｄｄという四種類は意味論的に考えたときにデータに対して不当な組み合わせでり、これらをデータフロー異常という。例えば「Ｕｒ」は、ある文で、変数に値が定義されない状態（未定義）のまま、参照されるというデータフロー異常を示している。

【００１４】図２は、データフロー異常が起こる実際のＣ言語によるプログラムの一例を示している。尚、ここでは説明の便宜上、各行の左側に行番号を付している。このプログラムにおいて、３行目のif文で、条件分岐があり、ｎが１より大きい場合だけ４行目で変数ｉに１というデータがセットされ（すなわち定義され）、６行目で１がセットされたこのｉが、ｊに代入（すなわち参照）されている。これに対して、３行目で、ｎが１と等しいか又は１より小さい場合には、変数ｉには１はセットされないが、６行目で、このデータがセットされていないｉがｊに代入されることになる。これは、上記の「Ｕｒ」というデータフロー異常に該当する。

【００１５】「Ｕｒ」以外のデータフロー異常は、制御構造上意図的に記述されることもあり得るので、データフロー異常のすべてがプログラムの誤りに関連するもの

ではない。たとえば、「Ｄｄ」は、データを参照しないで定義を二度繰り返した場合であり、「Ｕｕ」は、続けて未定義にする場合であり、「Ｄｕ」は、定義してすぐに未定義する場合であり、これらは実際上は無害である。しかし、例えば「Ｄｕ」の場合は、誤った変数名を参照しているといったプログラム上の誤りが潜んでいる可能性があり、また、これら三種類の組み合わせは、プログラム上必要となる記載でもない。したがって、ここでは「Ｕｒ」を含む上記の四種類の組み合わせをすべてデータフロー異常として取り扱う。

【００１６】データフロー異常検査器を用いてデータフロー異常検出を行うと、プログラムに対してデータフロー解析が実行され、まず、プログラムのどの文のどの変数についてどのような種類の異常があるかが検出される。本明細書では、これを「データフロー異常の検出」という。しかし、これだけだとデータフロー異常が起こっている場所が分かるだけである。実際のデバッグ作業等では、更に、このような異常がどのようなパスを通ったときに起こるかが分からなければならない。このため、一般のデータフロー異常検査器では、プログラムのどのようなパスを通ったときにそのようなデータフロー異常が起こるかを併せて調べる。本明細書では、これを「データフロー異常が起こるパスの探索」という。したがって、データフロー異常の検査といった場合には、通常「データフロー異常の検出」と「データフロー異常が起こるパスの探索」の両方を行う。

【００１７】しかし、このような仕方でデータフロー異常の検査を行うには、データフロー異常が起こるパスの探索の際にプログラム上のパスを一つずつ探さなければならない。例えば、ある文で変数に値が定義されないまま参照されるというデータフロー異常が起こった場合、データフロー異常検査器は、そのプログラムあるいは手続きの最初から、その文までのすべてのパスを調べて、異常が起こるパスを見つけ出す。このような作業を実行するのに要する時間は、検査対象となるプログラムの規模が大きくなるに従って、通常、急激に増大する。

【００１８】そこで、本実施形態のデータフロー異常検査装置では、検査対象であるソースプログラムを読み込むと、まず、後述する方法でそのパス数を計算で求め、この求めたパス数に基づいて、これ以降の動作を制御する。パス数が分かれば、そのソースプログラムについて行うデータフロー異常が起こるパスの探索に要する時間をある程度予測することができる。

【００１９】本実施形態では、パス数を計算するために、まず、読み込んだソースプログラムに基づいてフローグラフを作成する。フローグラフを自動的に描くソフトウェアは周知であり、本実施形態でもこのようなソフトウェアを使用する。図３は、Ｃ言語による簡単なソースプログラムの一例を示す図、図４はこのソースプログラムに対応するフローグラフを示した図である。フローグラフは、図４に示すように、ソースプログラムに含まれるif文のところで、パスが二つに分かれるように描かれる。尚、Ｃ言語には、if文以外にもソースプログラム中で分岐を生じさせる文が用意されているが、このような文はそれに等価なif文に変換してフローグラフを描く。図４に示したフローグラフは、元のソースプログラムが簡単なため、一見してパス数が三つであると分かるが、プログラムの規模が大きくなると、フローグラフだけからパス数を見積もることは困難である。

【００２０】そこで、フローグラフが描かれたら、図５に示すような制御依存グラフを作成し、この制御依存グラフを用いて所定の計算を行うことによって、パス数を見積もる。図５の制御依存グラフは、図４に示した各ノードについて、必ず通過するノードは、「関数ｆ」という最初のノード５１のすぐ下のレベルに描き、また、あるif文のノード、例えば図４のノード４３の実行結果によって、ノード４４或いはノード４５が実行されるかどうかが決まる時は、図５の制御依存グラフ上でノード５３からノード５４，５５へのパスが存在する。フローグラフ上でノード間に別のif文がないときは、制御依存グラフ上でノードとノードとの間に制御依存のパスを付ける。制御依存グラフのパスにはif文が真のときに実行されるのか、偽のときに実行されるのかというラベル、それぞれＴ，Ｆが付してある。

【００２１】制御依存グラフを描いたら、次の規則に従って、全体のパス数を見積もる。まず、すべてのノードに値１を付す。次に、条件文のノードを開始ノードから遠い順に並べる。具体的には、開始ノードから横形探索で順番付けしたノードの順番の逆順に並べかえる。図５では、条件文はノード５３とノード５５が存在しており、ノード５５の方がノード５３より開始ノードから遠いので、結果的には「ノード５５」、「ノード５３」、「ノード５１」の順番に並べる。この順番で各条件文のノードに対して、その条件文のノードにラベルＴで依存するすべてのノードの値を掛け算するとともに、ラベルＦで依存するすべてのノードの値を掛け算し、その二つの値を加算して得られた値をその条件文のノードの値とする。ラベルＴで依存するノードがなければラベルＴに対する計算値は１とする。ラベルＦも同様に行う。ノード５５においては、ラベルＴに値１（ノード５６）が一つ、ラベルＦにも値１（ノード５７）が一つ存在するので、

１＋１＝２

となり、ノード５５の値は２となる。同様に、ノード５３では、ラベルＴに値１（ノード５４）が一つ、ラベルＦには値２（ノード５５）が一つ存在するので、

１＋２＝３

となる。ついで、開始ノードであるノード５１は、ラベルＴに値１が二つ（ノード５２、ノード５８）、同じく値３（ノード５３）が一つ、ラベルＦには値１（ノード

５９）が一つ存在するので、
１×１×３＋１＝４
となる。

【００２２】すべての条件文のノードの処理が終わったら、開始ノードの値から１を引く。つまり、
４－１＝３
となる。これがそのソースプログラムのパス数となる。図５では、このようにして得られた値は３であり、図４に示したフローグラフのパス数と一致する。

【００２３】このようにして各ノードにおける値を求めたら、制御依存グラフの開始ノードの値から１を引いた値が、そのソースプログラムのパス数となる。次に、図６及び図７を参照して、本実施形態のデータフロー異常検査装置の動作について説明する。図６は、本実施形態のデータフロー異常検査装置のブロック図、図７は、この動作を示したフローチャートである。

【００２４】尚、図２に示したパス数設定手段２１、パス数見積もり手段２２、パス数比較手段２３、データフロー異常検査手段２４は、これらの機能をプログラムとして記憶媒体３１に記録したものをコンピュータ３０が読み取り、制御部３０ａの制御のもとでこれらのプログラムを実行することによって、コンピュータ３０上で実現することができる。したがって、この記憶媒体３１を例えばフレキシブルディスクやＣＤ－ＲＯＭ等の取り外し可能なものとすれば、この記憶媒体３１を配付することによって、この記憶媒体３１のプログラムを読み取って実行できる任意のコンピュータ上で本実施形態のデータフロー異常検査装置を実現することができる。

【００２５】図７のフローチャートにおいて、データフロー異常検査装置の動作を開始すると、まず、ユーザーがパス数設定手段２１によって所望のパス数を設定する（step１）。このパス数は、データフロー異常が起こるパスの探索を行うかどうかを決定する閾値となる。次に、パス数見積もり手段２２は、検査対象であるソースプログラム２０を入力する（step２）。検査対象であるソースプログラム２０が入力されると、まずこのソースプログラムに基づき、パス数見積もり手段２２によってフローグラフが作成される（step３）。フローグラフが作成されると、更にこれに基づいて、パス数見積もり手段２２が制御依存グラフを作成する（step４）。step４において制御依存グラフが作成されると、パス数見積もり手段２２はこの制御依存グラフに基づいて所定の計算を行って、このソースプログラムのパス数を見積もる（step５）。

【００２６】次に、パス数比較手段２３が、パス数見積もり手段２２により見積もられたパス数と、パス数設定手段２１により設定したパス数とを比較する（step６）。その結果、設定したパス数よりも少なければ、データフロー異常検査手段２４は、データフロー異常を生じる変数の検出とデータフロー異常が起こるパスの探索の両方を実行し（step７）、その結果を出力する。一方、設定したパス数と等しいか又はこれよりも多ければ、データフロー異常検査手段２４はデータフロー異常を生じる変数の検出は実行するが、データフロー異常が起こるパスの探索は実行しないで、そのプログラムのパス数及びデータフロー異常が起こるパスの探索を実行しない旨をディスプレー等のユーザーインターフェースに表示する（step８）。

【００２７】このように、予めデータフロー異常が起こるパスの探索を実行するかどうかを判定するためのパス数を設定し、検査対象であるプログラムのパス数がこれを超える場合にはデータフロー異常の検出は行うがデータフロー異常が起こるパスの探索は行わないようにすることによって、コンピュータが不必要にデータフロー異常の検査だけに専有されることを防止することができる。

【００２８】尚、本発明は、上記実施形態に限定されるものではなく、その要旨の範囲内で種々の変更が可能である。例えば、上記のように、予めパス数を設定してからプログラムのパス数を見積もる場合の他に、先に検査対象であるソースプログラムのパス数を見積もって、その値を見てから、オペレータがデータフロー異常が起こるパスの探索を実行するかどうかを判断するようにしてもよい。このようにすれば、検査対象であるソースプログラムのパス数が設定したパス数よりも多いが、その差がほんの僅かなため探索を実行する方が合理的であるという場合にも、柔軟に対応することができる。また、上記実施形態では、フローグラフからパス数を求める方法として、制御依存グラフを作成し、これに基づいてパス数を計算するという方法を採ったが、これ以外の周知の方法でパス数を見積もるようにしてもよい。

【００２９】更に、上記のように見積もったパス数を表示するだけでなく、得られたパス数から、そのパス数の場合にデータフロー異常の検査に要する時間を求め、その時間だけ、あるいはパス数と時間の両方を表示するようにしてもよい。

【００３０】また、本発明の各手段をコンピュータに行わせるように動作するプログラムを作成し、コンピュータに実行させることで、コンピュータを本発明のデータフロー異常検査装置として利用することができる。またこの場合、プログラムを記録媒体、例えば、ＣＤ－ＲＯＭ、フロッピーディスク、光磁気ディスク、磁気テープなどに記録することにより、当該記録媒体を読み込むことのできるコンピュータを容易に本発明のデータフロー異常検査装置として利用できることは言うまでもない。

【００３１】

【発明の効果】以上説明したように、本発明によれば、ソースプログラムを読み込んだ後、データフロー異常の検査を実行する前に、パス数見積もり手段によって、そのソースプログラムのパス数を見積もり、その結果に基

9

づいてデータフロー異常の検査手段の処理の実行を制御するので、例えばパス数が非常に多いためそのままパスの探索を実行すると、予定の時間を大幅に超えることが分かった場合には、例えばデータフロー異常の検出だけを行い、データフロー異常が起こるパスの探索を行わないようにしたり、または処理能力の高い別のワークステーションで検査を実行するなど、柔軟な対応が可能となり、作業を効率化することができる。

【００３２】また、予めオペレータが所定のパス数を設定しておき、パス数見積もり手段によって得られたパス数がこの所定の値以下の場合には前記データフロー異常検査手段によって前記ソースプログラムに対してデータフロー異常の検査を行うようにし、また、前記パス数が前記所定の値を超える場合にはデータフロー異常の検査のうちデータフロー異常の検出だけを行い、データフロー異常が起こるパスの探索は行わないように自動的に制御するようにしたことによって、パス数見積もり手段によるパス数の見積もりが終了した時点でオペレータが一々その後の動作を指示する必要がなくなり、操作性が向上する。

【００３３】更に、パス数見積もり手段によって前記ソースプログラムのパス数が得られたら、動作を一時停止し、そのパス数をユーザーインターフェースでオペレーターに知らせ、オペレータがこれを見てその後の動作を続行するかどうかを決定できるようにしたことによって、例えば、検査対象であるソースプログラムのパス数が設定したパス数よりも多いが、その差がほんの僅かなためデータフロー異常が起こるパスの探索を実行するのが合理的であるというような場合にも、柔軟に対応することができる。

10

【図面の簡単な説明】

【図１】ソースプログラムのデータフローを状態遷移図として示した図である。

【図２】データフロー異常が起こる実際のプログラムの一例を示した図である。

【図３】簡単なソースプログラムの一例を示した図である。

【図４】図３のソースプログラムに対応するフローグラフを示した図である。

【図５】制御依存グラフの一例を示した図である。

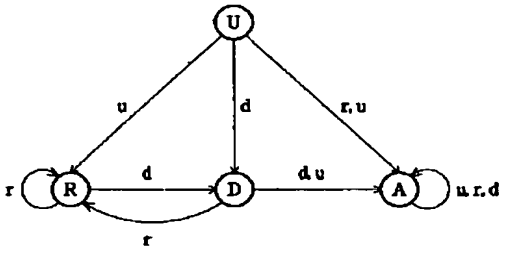
【図６】本発明の一実施形態であるデータフロー異常検査装置のブロック図である。

【図７】本発明の一実施形態であるデータフロー異常検査装置の動作を示したフローチャートである。

【符号の説明】

２０　ソースプログラム
２１　パス数設定手段
２２　パス数見積もり手段
２３　パス数比較手段
２４　データフロー異常検査手段
３０　コンピュータ
３０ａ　制御部
３１　記憶媒体

【図１】



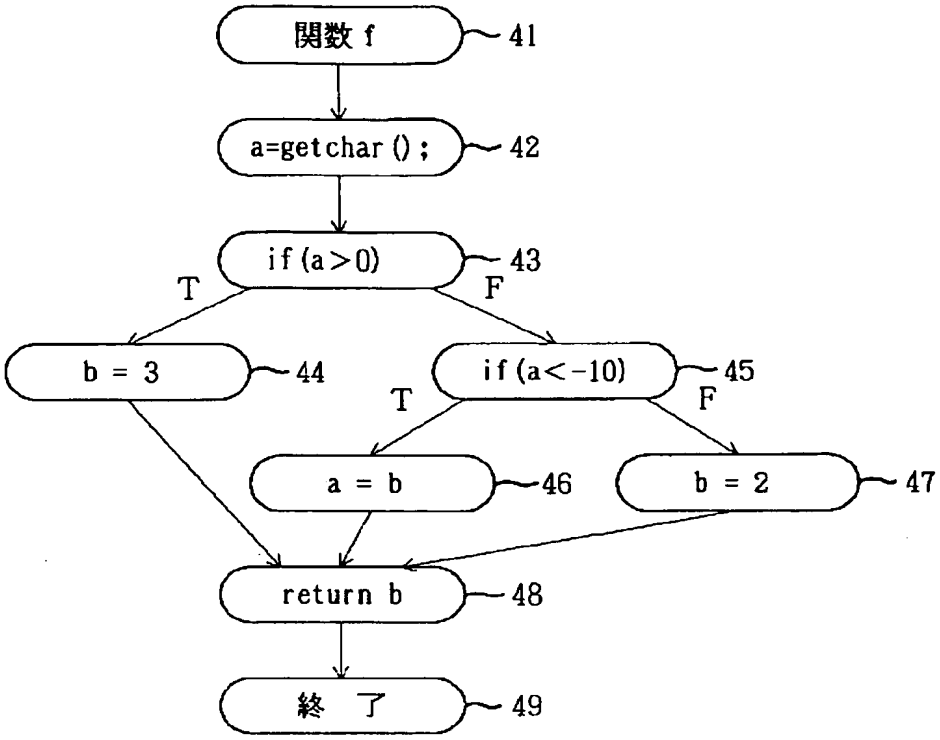
【図２】

```
1   int   f (int  n)
2   {     int  i, j;
3         if (n>1) {
4               i=1
5         }
6         j=i+1;
7         return  j;
8   }
```

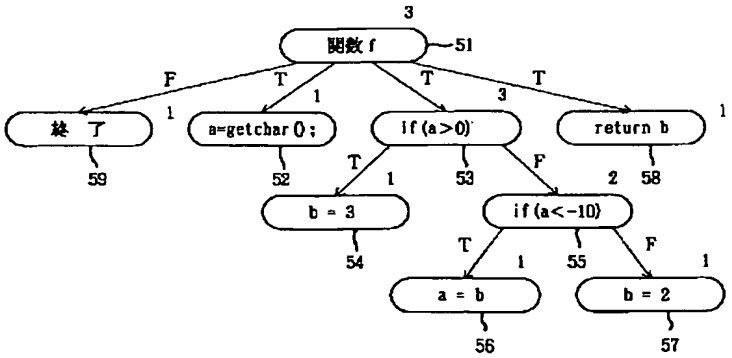
【図３】

```
int
f ()
{
   int  a, b;
   a=getcher () ;
   if  (a>0)  {
       b=3;
   }  else  {
       if  (a<-10)  {
           a=b;
       }  else  {
           b=2;
       }
   }
   return  b;
}
```

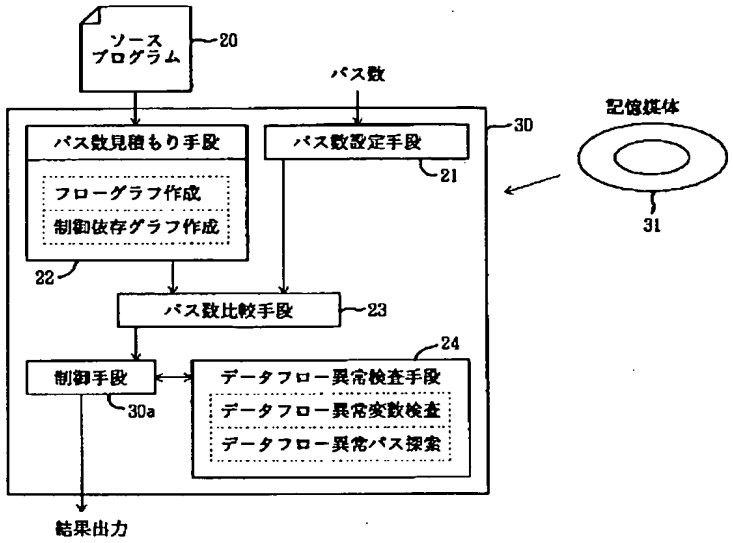
【図４】


関数 f
41
a=getchar ();
42
if (a>0)
43
T
F
b = 3
44
if (a<-10)
45
T
F
a = b
46
b = 2
47
return b
48
終 了
49

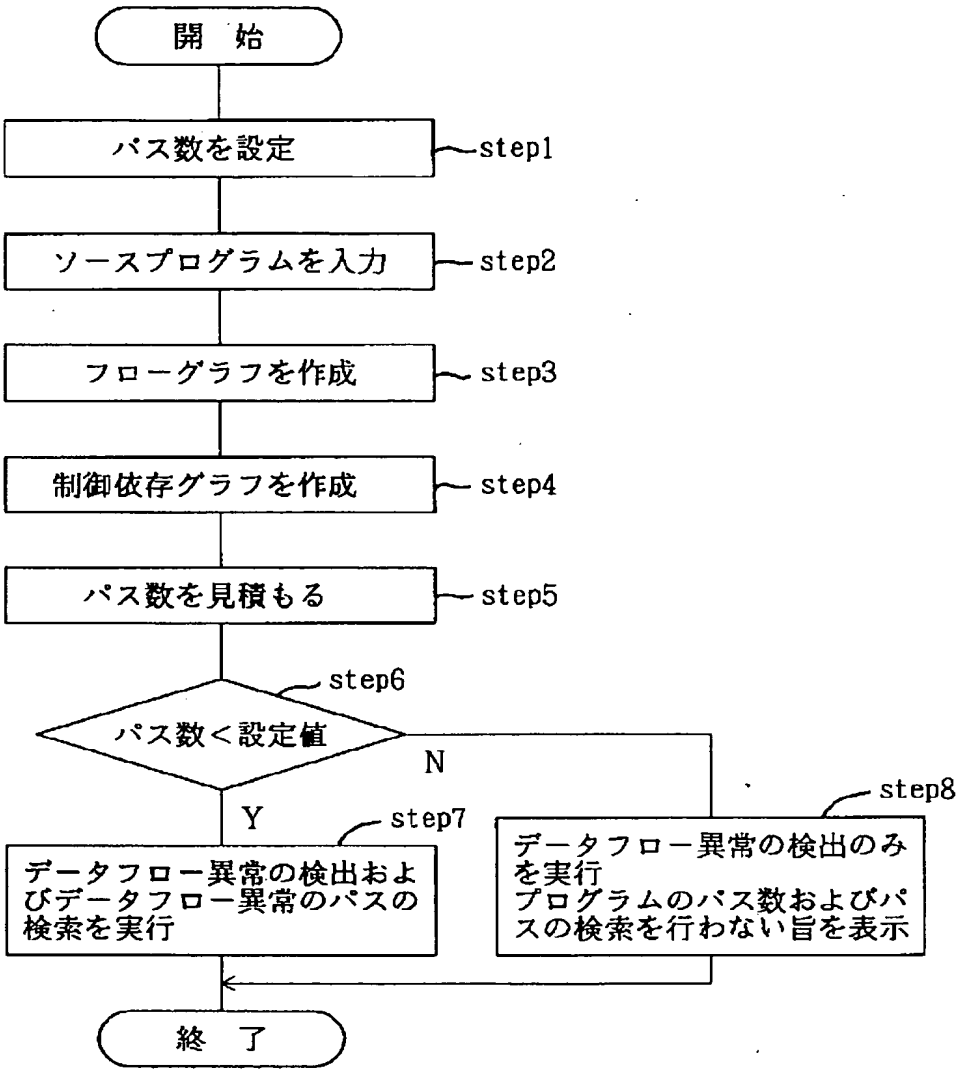
【図５】


3
関数 f
51
F
T
T
T
1
3
1
終 了
a=getchar ();
if (a>0)
return b
59
52
53
58
T
F
1
2
b = 3
if (a<-10)
54
T
F
55
1
1
a = b
b = 2
56
57

【図６】


ソース
プログラム
20
パス数
記憶媒体
30
パス数見積もり手段
パス数設定手段
21
フローグラフ作成
制御依存グラフ作成
22
31
パス数比較手段
23
24
制御手段
データフロー異常検査手段
30a
データフロー異常変数検査
データフロー異常パス探索
結果出力

【図７】


開　始
パス数を設定
step1
ソースプログラムを入力
step2
フローグラフを作成
step3
制御依存グラフを作成
step4
パス数を見積もる
step5
step6
パス数＜設定値
N
Y
step7
データフロー異常の検出およびデータフロー異常のパスの検索を実行
step8
データフロー異常の検出のみを実行
プログラムのパス数およびパスの検索を行わない旨を表示
終　了